Page No.
Date. / /

Java

Java is programming language and Platform

Developed by Sun Microsystems (Subdirectory of oracle) in 1995

James Gosling is known as father of Java

```
Class simple {

    Public Static Void main (String args[])
    {
        System.out.println ("Hello Java"); }
}
```

Java has Runtime Environment and API hence it is called Platform as well

it is Statistically typed

(WORA)
Write once Run Anywhere

Applications

1. Standalone
2. web Applications
3. Enterprise Applications (distributed Banking, EJB
4. Mobile Applications

Platforms.

1. Java SE : Programming Platform, Core Java
2. Java EE : web Applications (servlet, JSP)
3. Java ME : Mobile Applications.
4. Java Fx : Rich internet Applications.
5. Java Cards

JVM, JRE, and JDK are Platform dependent.
However Java is platform independent.

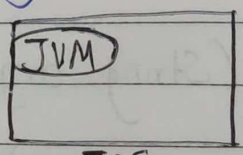
Page No.
Date.

* JVM (Java Virtual Machine)

Provides Runtime Environment to Javabytecode.
load → verifies → executes → provide Runtime Environment

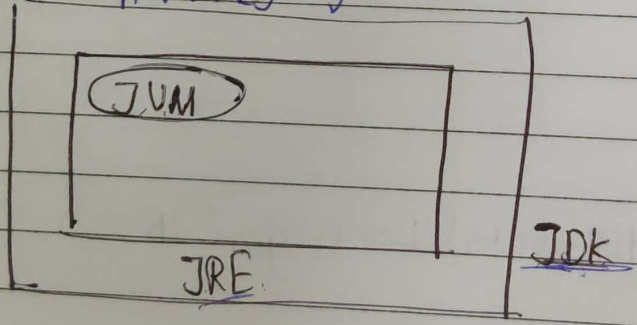
* JRE (Java Runtime Environment) or Java RTE

This is physically exists, A Implementation of JVM

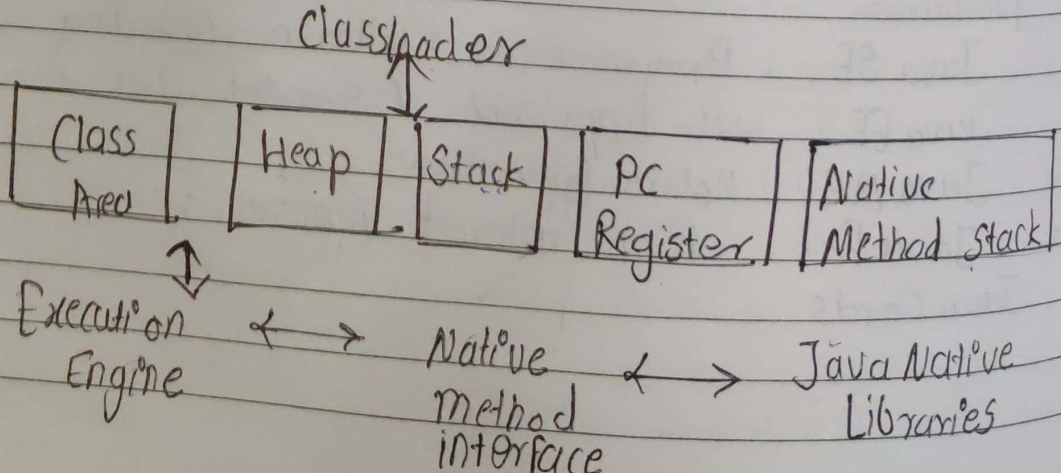


* JDK (Java Developement kit)

Java interpreter/loader (Java), Java compiler (Javac) and an archiver (jar)



1) JVM



Page No.
Date. / /

1 Classloader

Bootstrap Classloader (All class files)
↓
Extension classloader ($ Java Home /jre/lib/ext)
↓
System / Application classloader (classpath)

2 Class (Method) Area : Preclass Structure

3 Heap : Runtime data area in which objectes are allocated.

4 Stack : local variables & partial results, Each thread has private JVM stack
Java Stack Stores frames, ~~each~~ new frame is created each time method is invoked and distroyed when method invocation Completes.

5 Program CounterRegister (PC) : address of Jvm instruction currently being executed.

6 Execution Engine:
1) Virtual processor
2) Interpreter : Read bytecode Stream → execute
3) Just In Time (JIT) compiler:
improve Performance. bytecode having Similar functionalities compiles

7 Native Method Stack
8 Java Native Interface (JNI)
framework which provides interface to Communicate with other Applications written in other languages

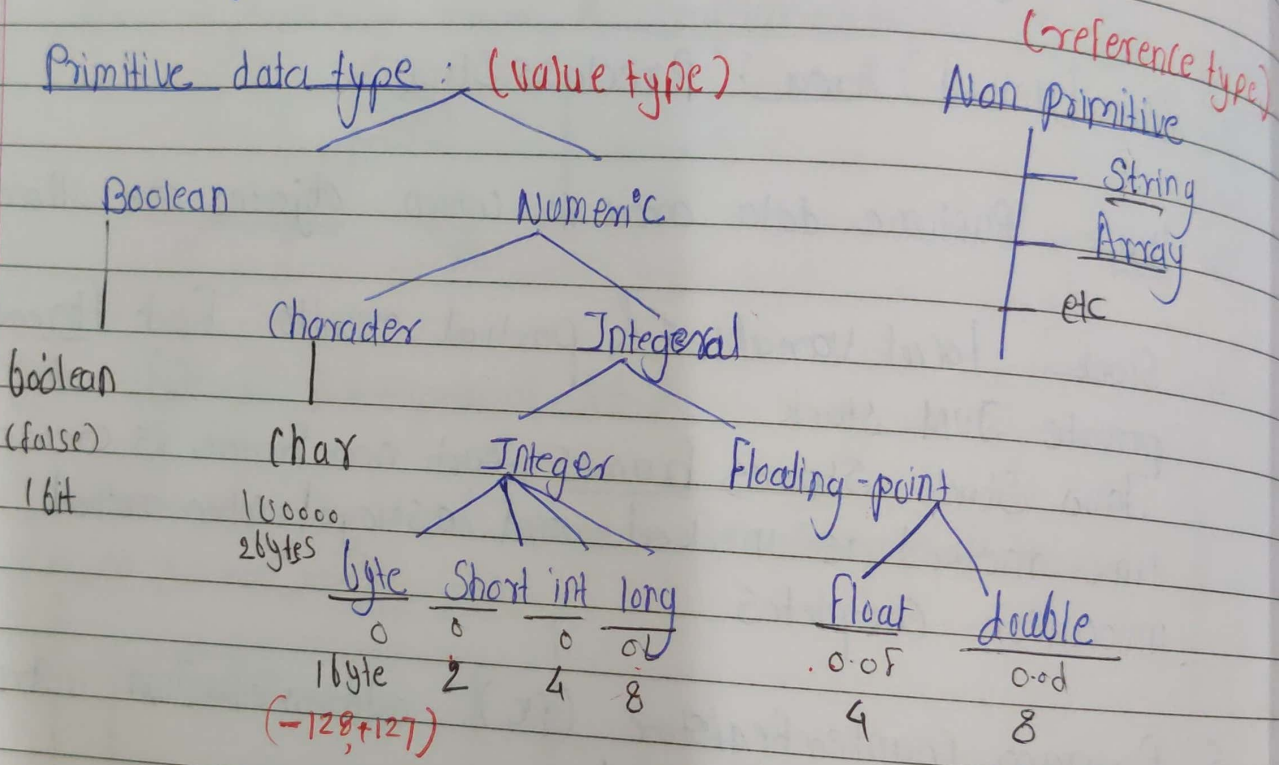
Java is statically typed programming language
means all the variable must declared before it's use

Page No.
Date. / /

★ Java Variables

local - inside method
Instance - within class outside method, instance specific & not shared among instances
Static - it can't be local, create single copy & share it among all the instances. Memory allocation happens only once

★ Java Datatypes



★ Unicode System

Java Uses Unicode System. in unicode Char = 2byte
hence in Java Character holds 2 bytes
lowest value = \u0000
the highest value = \uFFFF

★ Operators in Java

Unitary operators
x++ , ++x , x++ , x--

Page No.
Date. / /

```
Public class Operator Example
{
Public Static Void main (String args[])
int x=10;
System.out.println (x++)    // 10 (11)
"                  (++x)    // 12
"                  (x--)    // 12 (11)
"                  (--x)    // 10   }
```

eg.
a=10 and b=10

a++ + ++a ⇒ 10 + 12 ⇒ 22
b++ + b++ ⇒ 10 + 11 ⇒ 21

eg.
int a=10, int b=-10
(-a) ⇒ -11 and (-b) ⇒ 9
positive value asel tr Plus 1 ani sign -
Negative value asel tr -1 and sign +

(10*(10/5)+3-1*(4/2)) ⇒
(10*(2) +(3-1) * 2)
20 + 42 = 24

(10*10/5+3-1*4/2)

10*2+3-1*2

20+3-2
~~20+3~~ 23-2

21

BODMAS
(Bracket)
Division
multiplication
addition/subtraction

Page No.
Date.

10 = 1010
left = 1010

2) Shift operator:

left shift → (10 << 2) ⇒ 10 * 2^2 ⇒ 10 * 4 ⇒ 40

right shift (10 >> 2) ⇒ 10 / 2^2 ⇒ 10 / 4 ⇒ 2

>>> works same for positive
>>> changes parity bit (MSB to 0) (First bit of binary)
(-20 >>> 2) = 1073741819

3 logical operator

logical and :- it doesn't check for second condition when first condition is ~~true~~ false (&&)
Bitwise(&) checks both regardless of T or F

logical OR :- it doesn't check for second condition when first condition is True (||)
Bitwise(|) checks both regardless of First Condition T or F

4) Ternary operator: (conditional operator)
if then else rule, it takes 3 operands

a = 2
b = 5
(a<b) ? a:b ⇒ 2

a = 10
b = 5
(a<b) ? a:b ⇒ 5

5) Assignment operator

a+=b // a = a+b

Short a=10;
Short b=10;
a=a+b ⇒ (10+10 = 20)

Compile time error because Now it is in integer
After typecast it will give output

Page No.
Date. / /

* keywords = ~~356~~ Above 50

enum :- Set of constants, enum constructor are always private or default.

final :- final is keyword & finally : block of code in try-catch str

* Control Statement in Java

Decision Making Statements
- if statements [if, if else, if-else-if, nested if]
- Switch statements

loop Statements
- do while loop
- While loop
- for loop
- for each loop = int arr [] = {1,2,3,4}
  for(int i, arr) { }.
- labbled for loop

Jump Statement
- break Statement (Exit from innermost)
- Continue Statement (Skip next iteration without finishing current)

diff. between C++ & Java

Java doesn't support operator overloading, multiple inheritance, pointers, header file. files, ::, unsigned integers
Java has method overloading but not operater overloading

* Comments

Implementation Comment /* ... */ and //

Documentation Comment /** ... */

* Entry point method

JVM invokes main method. We can overload main method in Java

* Wrapper Class

In Java primitive types are not classes. But for every primitive type Java has defined a class called Wrapper class

All are final & declared in Java.lang package

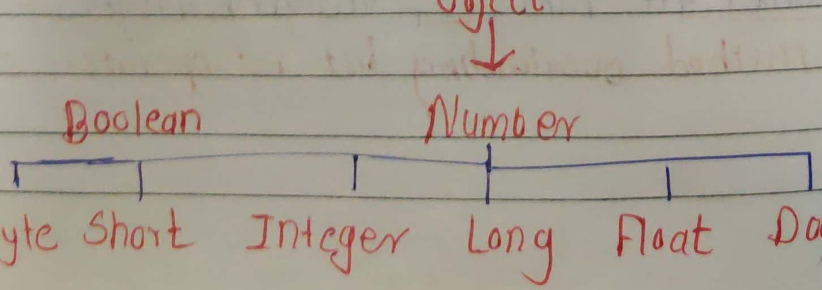
Uses

1 To parse String (To Convert String to numeric type

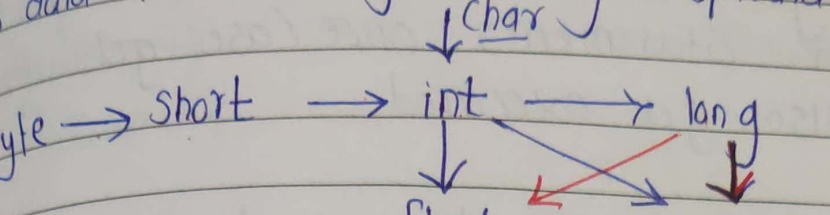
```
int num = Integer.parseInt("123")
```

2 To Store primitive type value into instance of generic class, type argument must be a wrapper class

```
Stack<Integer> stk = new Stack<Integer>();
```



* Widening
Converting int to double // small to large conversion
No data loss so type casting is optional



int → long → Float → double

* Narrowing (Forced Conversion)

Wider → Narrower // data loss
Explicit casting is mandatory
double n = 10.5;
int n2 = (int) num1; # Reverse flow of widening ↑
Both for primitive

* Boxing
primitive → Non primitive

- Integer.toString()
- String.valueOf()

* Unboxing
Integer.parseInt() Non primitive → primitive

if String doesn't contain numbers then parsexxx() throws exception

- Integer.parseInt()
- Float.parseFloat()
- Double.parseDouble()

Page No.
Date. / /

★ Switch expression

Switch fallthrough = when we don't use break Statemen
then all other statements once case gets true, all
other will also get executed.

enum = to keep constants throughout

★ infinitive for loop

for(;;) { }

for loop	While loop	do while [Exit Cont
When No. of iterations are fixed.	When not fixed	when not fixed & we have to execute loop at least once
infinitive Syntax for(;;)	while(true) { }	do { } while (true);

★ Scanner
text based parser

public String nextLine()
Public int NextInt()
Float NextFloat()
double nextDouble()
} Methods of & Scanner Cl

★ Java Buzzwords (features)

1) Simple 2) object Oriented 3) Architecture Neutral 4) Portable
5) Robust 6) Multithreaded 7) Dynamic 8) Secure
9) High Performance 10) Distributed

Page No.
Date. / /

Major pillars of oops
Abstraction
Encapsulation
Modularity
Hierarchy

Minor
Typing / polymorphism
Concurrency
Persistence

Multithread
JVM executes threads
Main thread / User / Non daemon (5) default priority
Garbage collector / Daemon / background (8) (5+3)

Access Modifiers

	Same class	Sub class	No sub class	Different package Sub Class	Different package Non sub class
	✓	×	×	×	×
ge level private default)	✓	✓	✓	×	×
ected	✓	✓	✓	✓	×
blic	✓	✓	✓	✓	✓

- get memory in heap

* object :- instance of the class, when object is created constructor invokes automatically, Real world entity, runtime entity
Every instance on heap is Anonymous

* Class
Variable inside class called field or Attributes
Methods = operation behaviour / message

* Instance Variable :- inside class but outside method doesn't get memory at compile time, get at runtime

★ This keyword
Reference Variable
- can be used as instance variable
- invoke Current class method
- " " " Constructor
- Passed as an Argument in method call
- " " " " Constructor call

when we pass
this (name, Roll, course), this Should be first statement in

★ Constructor
- Name is same as class name
- doen't have any return type
- called once

Everytime when object is Created using new() keyword at least one Constructor is Called
No need to Create Class's Constructor, Compiler Create it

Parameterless Constuctors / Default
- Used to provide default value

Parameterized Constructor
- different values to distinct object

- Constructors can be overloaded
- There is No Copy Constructor

★ ways to Copy Value of one Object to another
- By Constructor
- by Assigning value of one object to another
- Clone() method of Object Class

Page No.
Date. / /

- Constructor can return value: ,
- Constructor can create object, start thread, call method, it is also class in Java
- We can call constructor from another constructor called constructor chaining

* Literals
int num = null ; //invalid ; // Integer num = null //valid

* oops
Simula , first oop , Smalltalk - first truely oops

- object is a real world entity
- class is collection of object

Inheritance - Parent → child (is-a)

Polymorphism - task perform in different ways

Abstraction - Hiding details & showing functionality eg. phone call

Encapsulation - wrapping data together

Coupling - info dependency of another class

Cohesion - Component which perform single well defined task
Java.io is highly cohesive, Java.util is weakly cohesive

Association : Relation between objects (1-1) (1-M) (M-1) (M-M)

Aggregation - (Has-a) way to achieve Association
Another way to reuse object, Weak relationship

Composition :- Also way to achieve Association
strong relationship
Parent obj deleted, Child will be deleted

★ Literals
Constant value appear directly in program

1) integer → i) Decimal = eg. 26, 5678
2) octal = 045, 026, 076
3) Hexa decimal = 0x1a (Start with x0 or X0)
4) Binary = 0b11010
5) Real = 879.0, 99E-3 (Fractional)
6) Backslash = \n, \t, \b, \v (vertical)
Horizontal ↓ blank
\a - small beap, \r = return, \', \"

2) Character : 'a'

2) String literals : " "

3) Floating Point literals - 6f, 8.34f, f - float type
6d, 835D - double type

4) Boolean :- true, false, 0, 1

5) Null
6) Class - Scanner.class
7) Invalid - 6_6.74f9, 77_; 0x 12;

Page No.
Date. / /

* Naming Conventions

Class - Public Class Employee
Interface - interface printable
Method - draw()
variable - int id;
Constant - MIN_AGE = 18;

* ways to Create object in Java
- Newkeyword (new)
- ## newinstance()
- Clone()
- deserialization
- factory method

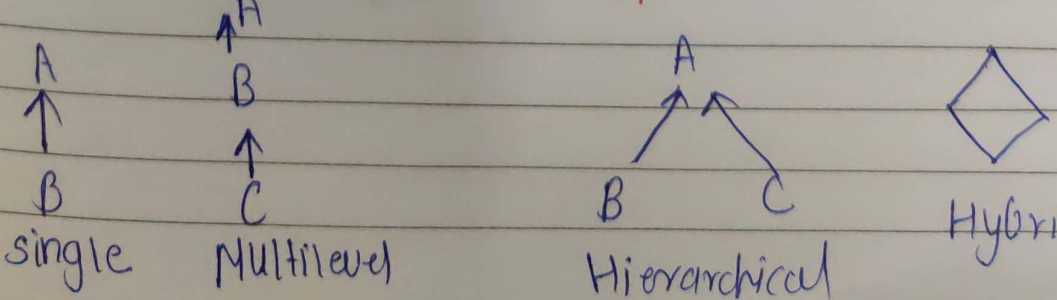
if we wants to use object only once then anonymous object created

* Static Keyword
Memory Management

Variable, Method, Block - execute before main()

* Inheritance
code reusability, Method overriding
class subclass extends Superclass-name { }

Multiple inheritance doesn't Support in Java



Date. / /

* Aggrigation
Code reuse is best achieved by aggregation when there is no is-a relationship.

* Method overloading
Same name different parameters

it is not possible by Changing return type of method only
we Can overload main method, but JVM recieves (String [] args) only

Static double add (int a, int b) {return a+b} *

type promotion → int - double
de-promotion Can't be done implicitly

* Method overriding

When child class has Same method as declared in parent
Runtime polymorphism
Same name Same parameters

Static methods Can't ~~be~~ override (main)

* Covarient Return type

Methods with different return type Can be override

Page No.
Date. / /

★ super keyword

immediate parent class instance variable.
" " " method.
" " " constructor.

★ Instant Initializer Block

it get invoked ~~when~~ at the time of object creation.
First ~~Cons~~ Constructor gets invoked then instance initializer block get invoke

★ final
Variable, method, class ⟶ can't extend it
↓ ↓
Never be changed Never be override

- final can be inherited. Blank final means uninitialised and blank final variable initialized only in constructor.
- Static blank final variable, it can be initialized in static block
- final parameter - we can't assign value to final variable
- Constructor can't be final

★ Polymorphism

Compile Time Polymorphism
- Method overloading

Runtime polymorphism
Dynamic method Dispatch
overriding

Upcasting :- Reference variable of parent class refers to object of child class

ClassA Aa = new B( ) ClassB

- Runtime ~~polymirphi~~ ~~ff~~ polymorphism can't be achieved by ~~data~~ data members

★ Static Binding & Dynamic Binding
Early late

Binding - Connecting a method call to the method body

Static - object type is determined at compile time
dynamic - " " " " " Runtime

Private, final or static method - Static binding

★ instance of
Type Comparison operator
- it returns true or false, when applies to null it will retur
false.

★ downCasting

Page No.
Date. / /

Dog d = new Animal() // Compilation error
Dog d = (Dog) new Animal() // it will compile but Class Cast Exception

* Abstract class:
ways to achieve abstraction
1 Abstract class (0 to 100%)
2 interface (100%)

- Must declared with abstract keyword
- Can have abstract & non abstract methods
- it can't be instantiated
- it can have final methods
- it can have constructors and static methods also.

- if there is an abstract method in the class, class should abstract.

* Interface in Java

- Blueprint of a class Static constants and abstract methods
- Mechanism to achieve abstraction and Multiple inheritance.
- it can't have method body.
(is-A) Relationship
- Java 8 have default & static method in interface
Java 9 have private methods in interface.

- To Achieve Abstraction
- Multiple Inheritance
- To achieve loose coupling.

interface <interface_name> { }

- Compiler adds public & Abstract keywords before interface method, it also adds public, static, final

class	interface	interface
↑ extends	↑ implements	↑ extends
class	class	interface

Multiple inheritance in Java achieved by interface

- Marker / tagged - when there is no member is known as marker eg Serializable, Clonable, Remote etc

★ Packages in Java

default package = Java.lang
if we import package, subpackages will no be imported

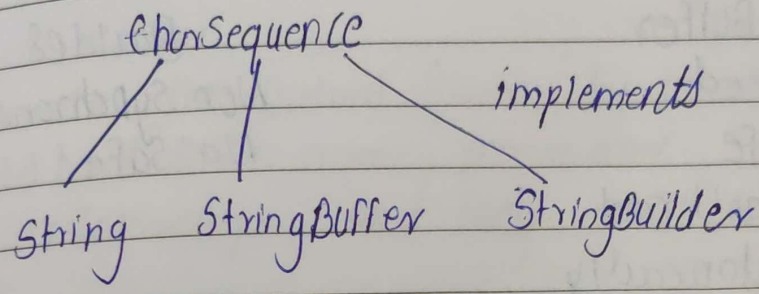
domain.company.Package
Com.Javapoint.bean

★ Encapsulation:
Java Bean is fully encapsulated class

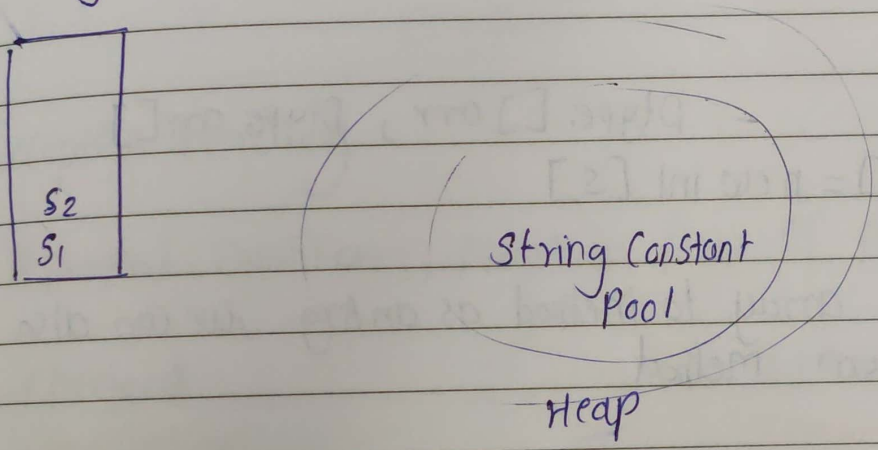
Page No.
Date. / /

★ String

Java.lang.String implements Serializable, Comparable and CharSequence interfaces



String S1 = "Hi" and String S = new String ("Hi")



immutable in Java

★ StringBuffer:- Strings can be modified ie mutable strings can be achieved by StringBuffer class

StringBuffer capacity() - Default is 16
(16*2)+2 = 34

★ StringBuilder:- mutable String, it is Non synchronized

String
Immutable
String Constant pool

StringBuffer
mutable
Heap

StringBuffer
- Synchronized
- Thread Safe
- 2 threads can't call method simultaneously
- less efficient

Builder
Non Synchronized
No Safe

★ Arrays
1D - Dtype [] arr, Dtype arr[]
Int a [] = new int [5]

we can pass array to method as an Arg. , we can also return array from method

Multidimentional

int [][] arr = new int [3] [3]

★ Jagged Array in Java

2D array with odd No. of columns
int arr [] [] = new int [3] [];

★ System.arrayCopy ()

Page No.
Date. / /

* Cloning of Arrays

1D - deep copy - Copy Actual value
Multidimentional - Shallow Copy - Copies references

* Virtual function

- override the virtual function by inhering class fun
- Should have same name, parameter
- is-a relationship and they can't be private, Final, Static

By default every Non-Static method is Virtual function

* Exception Handling

Abnormal condition, Mantain flow of Application

Checked
Compile time
inherit throwable class
IoE SQL

Unchecked
Checked at Runtime
inherit RuntimeException
Arithmetic, Null point

Try
Catch
Finally
Throw
Throws

int - long - boxing
long - int - unboxing

Page No.
Date. / /

★ Enum
Fixed set of constants
All have capital letters, static and final implicitly
Enum class
improve type safety, easily used in switch, can be traversed, feids, constructors and methods
Enum can't extend Class

Compiler internally adds values(), valueof() and ordinal() methods within the enum at compile time, it internally creates static and final class

★ ~~Autoboxing~~
Primitive → wrapper manually not required

Unboxing - wrapper (Integer) - int

with method Overloading

Widening beats boxings - int → Short → int ✓
Short → Integer ✗
widening beats varargs 2 Short → int int ✓
Boxing beats varargs int → Integer

Method overloading with widening & Boxing
int → long ✗

★ Annotations:-
Metadata
@ Override @ Suppress Warnings @ Deprecated
@ Target @ Retention @ Inherited @ Documented

Custom Annotation

```
@interface MyAnnotation {}
@interface //
int value(); }
```

Marker – No method
SingleValue – one method
Multi value

API [Application Programming Interface]

Computer Application's interface

Generics < >

type Safe objects ; makes code Stable and detecting bugs at compiletime

Type Safety
Type Casting not required
Compile Time Checking

Generic Class
type parameters T E k N V
Type Element key Number Value

Wildcard = ?
Upper bond wildcards List < ? extends Number>

Unbounded : List <?> - don't depend on type para

lower bounded List < ? super Integer>

Page No.
Date. / /

★ Transmission Control protocol (TCP) & Internet protocol

User Datagram P – offline

TCP/IP – 1024 holds ports
23 – Telnet , 21 – FTP , 43 – Whois
25 – Email , 80 HTTP , 79 – finger
119 – Netnews

Stream Sockets	Datagram Sockets	Raw Sockets
Connection ori	UDP	ICMP
Sock_Stream	Sock_DGRAM	SOCK_RAW
TCP	↔ bidirection	IPv4 & IPv6
Client Server		Connectionless

★ Functional Interface
@ functionalinterface
Interface Contain exactly one abstract Method
SAM

★ Java.util.function package

Predicate <T> = Boolean Value function of one Argument
takes Argument & produce Boolean

Map =

Consumer = Consumes Value , Returns Nothing

Supplier = produces result

Page No.
Date. / /

* Lambda Expression
implementation of interface Functional
less coding

(Argument-list) → {body}

can have 0 or n No. of Arguments

* Effects of functional programming on Collection

Stream, lambda, filter, map, & reduce
powerful, efficient, robust

* Stream
Java.Util.Stream
classes, interfaces, enum

doesn't store elements, conveys elements from source through a pipeline of computational operations
functional, doesn't change source
lazy, evaluate when required
iterate only once

- filter
- map
- collect

Page No.
Date. / /

Stream	Collections
- doen't store data Use function interface	Stores data
- Consumable	Non-consu., traverse many ti
- Sequential & parallel	Parallel
- Java.util.Stream	intstream, Longstream, Doublestream
- Not Modifiable	

★ Stream API

intstream — IntStream.of() , intstream Stream()
map() , filter() , sum() , max() , min()

longStream

doublestream
of() - Single element
filter - returns Stream elements that match given predica

map - Transformed by mapper

reduce - Reduce the elements

Sort -

flatMap - flatten element

Anymatch

Page No.
Date. / /

Count

Boxing - Boxed to int

Java Concurrency

Two ways to create threads in Java
By extending thread class or by implementing Runnable interface
override run() method
Run() implement, No need to override

Lifecycle of Thread

New
Active - runnable, running
Blocked / waiting
Timed waiting - Starvation Sleep()
Termination

Thread.State

Advantage & issues

Improve performance
Responsiveness
Resource Utilization

Complex
Race condition - Same time accessing
Deadlock - waiting for each other

Page No.
Date. / /

★ Threadgroup in Java

Java.lang.ThreadGroup

Set of threads, other thread group
it's like a tree,

★ Java Runnable Interface

Used to execute code on a concurrent thread
~~the~~ when we want to run only run method

★ Synchronization in Java

it is capability to control the access of multiple thread to any shared resource

when we want to allow only one thread to access the shared resource

Prevent thread interference, To prevent consistency

Process Synchronization
Thread Synchronization

- Mutual Exclusive
  1) Synchronized method - lock
  2) block - 50-lines / 5-lines-block, efficient
  3 Static Synchronization - lock on class
- Inter thread Communication
  - wait()
  - Notify()
  - NotifyAll()

Page No.
Date. / /

wait
Release lock
object class
Non static

Sleep

★ Executor framework

Bunch of components that helps to manage thread efficiently

Single thread Executor - single thread, sequential order
fixed thread Pool (n)
Cached thread pool - short live parallel
Scheduled Executor

★ Java io package

Stream - sequence of data, composed of bytes
System.out, System.in, System.err
write() read()

★ Byte stream & Unicode Character Stream

Bytestream - read bytes, Audio, Video (8)

inputStream outputStream
BufferedOutputStream

Character Stream (16 bits)
Reader Writer
BufferedReader
FileReader

Page No.
Date. / /

* Persistent object & Transient object

Persistent object → Continue to exist in database
Storing object for later use

Transient object → exist only within memory Until process exists

* Serialization & Deserialization in Java

State of an object → Bytestream
Hibernate, RMI, JPA, EJB, JMS
writeobject() : Readobject()

Serializable is Marker interface (No member)
Clonable & Remote are also marker
String & wrapper uses Serializable default

Methods
Public Final Void writeobject
" " flush()
close()

* Reflection

Examining or Modifying Run time behaviour of a class at run time
lang.reflect

IDE , eclipse, NetBeans
Debugger
Test tools

Java.lang.Class
- used to get metadata

3 ways to get instance of class
forName()
getClass()
the.Class gg Syntax

inspection is possible So avoid error
Cons:-
we can access private methods
Security Compromised.

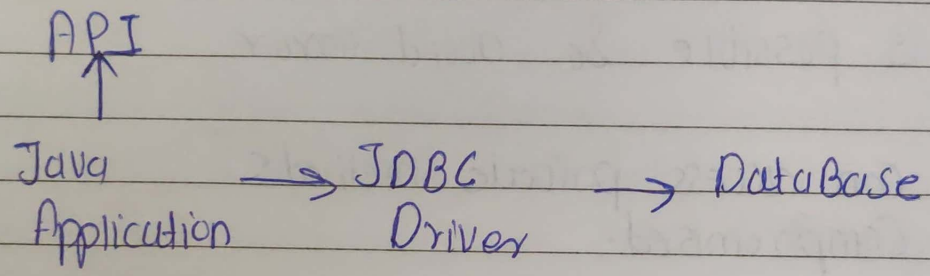
- Dynamic method invocation. is ability to call method on object without knowing specific class at compile time
Achieved using reflection
- getmethod() → invoke()

- Call objects that are created dynamically
" different classes
not known at compile time

JDBC Java Database Connectivity

Types of Driver

1. JDBC-ODBC
2. Native
3. Network protocal
4. Thin Driver



Java.sql - JDBC API

Popular interfaces= Driver, Connection, Statement etc

Classes= DriverManager, Blob class, Clob class, Types class